02

Requisições Ajax com o objeto XMLHttpRequest

Transcrição

Nosso objetivo é que ao clicarmos em "Importar Negociações", seja chamada uma ação na `controller`. Buscaremos o trecho referente no código do `index.html`.

```
<div class="text-center">
    <button class="btn btn-primary text-center" type="button">
        Importar Negociações
    </button>
    <button class="btn btn-primary text-center" type="button">
        Apagar
    </button>
</div>
```

Dentro da tag `<button>`, vamos adicionar o `onclick`.

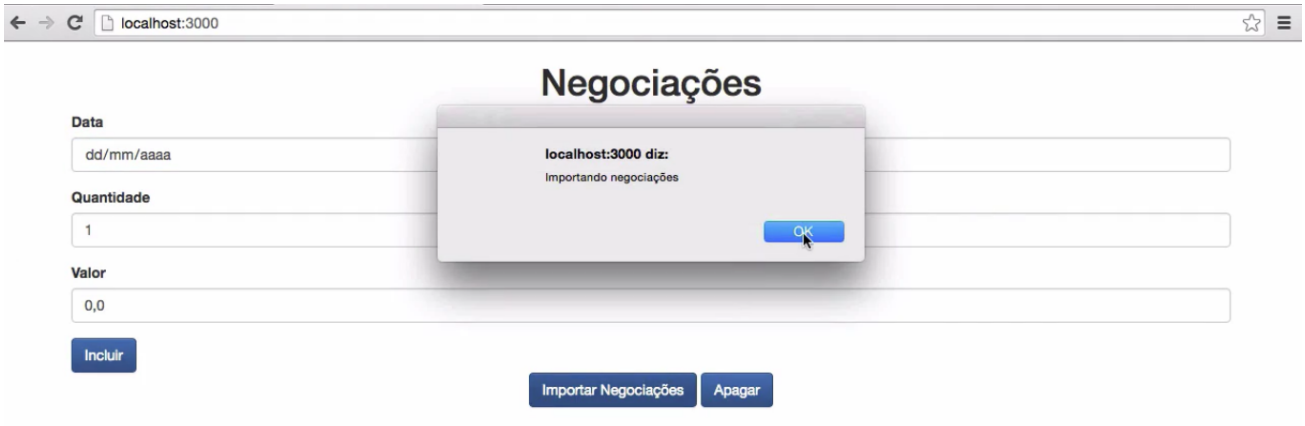
```
<div class="text-center">
    <button onclick="negociacaoController.importaNegociacoes()" class="btn btn-primary text-cent
        Importar Negociações
    </button>
    <button class="btn btn-primary text-center" type="button">
        Apagar
    </button>
</div>
```

Nós chamamos o `importaNegociacoes()`, que será adicionado no `NegociacaoController.js`, logo abaixo do método `adiciona()`.

```
importaNegociacoes() {

    alert('Importando negociações');
}
```

Desta forma, poderemos ver se tudo está funcionando bem. Se recarregarmos a página no navegador e depois, clicarmos em "Importar Negociações", o `alert` será exibido.



Agora, realizaremos requisições assíncronas para o servidor usando o objeto especial existente no JavaScript. Substituiremos o `alert` pela variável `xhr`.

```
importaNegociacoes() {
    let xhr = new XMLHttpRequest();
}
```

A variável é uma instância de `new XMLHttpRequest()`. É comum vermos desenvolvedores realizarem o AJAX utilizando apenas jQuery, mas nós faremos manualmente. Pediremos para `xhr` abrir um endereço e especificaremos qual método será utilizado:

```
importaNegociacoes() {
    let xhr = new XMLHttpRequest();
    xhr.open('GET', 'negociacoes/semana');
}
```

O método `open()` recebeu dois parâmetros: o primeiro especifica o tipo de requisição a ser realizada(`GET`), o segundo é o endereço (`negociacoes/semana`). Se trabalhássemos com outro endereço do serviço na Web, seria necessário colocar o endereço completo. Como estamos trabalhando localmente, só adicionamos `negociacoes/semana`.

Mas até aqui, a requisição não será executada. Para que a ação seja executada, usaremos o método `send()`.

```
importaNegociacoes() {
    let xhr = new XMLHttpRequest();

    /* configurações */
    xhr.open('GET', 'negociacoes/semana');

    /* executa */
    xhr.send();
}
```

A requisição ainda não está pronta. Será preciso fazer várias configuração antes de realizar o envio. É o que faremos a seguir: primeiramente, precisamos entender que toda requisição AJAX passa por estados - um deles nos dará os dados retornados do servidor. Por isso, precisamos interagir com esses estados e especificar que adicionaremos os dados de um deles no nosso `model`. O `xhr` tem a propriedade `onreadystatechange`, depois, passaremos uma *arrow funtion* que será chamada sempre que o estado do `xhr` for modificado.

Então, quais são os estados possíveis de um requisição AJAX? Listaremos abaixo os estados:

- 0: requisição ainda não iniciada
- 1: conexão com o servidor estabelecida
- 2: requisição recebida
- 3: processando requisição
- 4: requisição está concluída e a resposta está pronta

O estado **4** é o que mais nos interessa, porque é nele que temos acesso à resposta enviada pelo servidor. Precisamos encontrar um forma de descobrir em qual estado estamos quando o `onreadystatechange` é chamado. Para isto, adicionaremos um `if`:

```
importaNegociacoes() {

    let xhr = new XMLHttpRequest();
    xhr.open('GET', 'negociacoes/semana');
    xhr.onreadystatechange = () => {
        if(xhr.readyState == 4) {

        }
    };
    xhr.send();
}
```

Se o estado for igual a `4`, buscaremos os dados do servidor. Este é o estado que a requisição está concluída e a resposta está pronta, mas não podemos confiar no mesmo, porque pode ocorrer algum erro no servidor e ainda assim, a resposta será válida. É comum o servidor lidar com estados da requisição. Então, só teremos certeza de que os dados chegaram se o `xhr.status` for igual a `200` (*status code* genérico).

```
if(xhr.status == 200) {
}
```

Se fosse um status de erro, poderíamos colocar `400` ou `500` ... Usaremos os dois `if`s para nos blindarmos de qualquer tipo de problema.

Com as modificações, o nosso código ficará assim:

```
importaNegociacoes() {

    let xhr = new XMLHttpRequest();
    xhr.open('GET', 'negociacoes/semana');
    xhr.onreadystatechange = () => {
        if(xhr.readyState == 4) {
          if(xhr.status == 200) {
          }
        }
    };
    xhr.send();
}
```

A seguir, faremos um teste, adicionando o `else` e o `console.log`.

```
importaNegociacoes() {

    let xhr = new XMLHttpRequest();
    xhr.open('GET', 'negociacoes/semana');
    xhr.onreadystatechange = () => {
        if(xhr.readyState == 4) {
            if(xhr.status == 200) {
                console.log('Obtendo as negociações do servidor.')
            } else {
                console.log('Não foi possível obter as negociações do servidor.')
            }
        }
    }
    xhr.send();
}
```

Será exibida uma mensagem de erro quando o valor for diferente de `200`. A questão é: como teremos acesso às requisições que vieram do servidor?